NAME

shapes − simulate scattering of 2d electromagnetic (E_x, E_y, H_z) waves and pulses on metal objects of various shapes and with various material properties

SYNOPSIS

Parallel version: **mpirun** −np *number-of-processes* −machinefile *node-file* **pshapes** *shapes-input-file*

Sequential version: **shapes** *shapes-input-file*

DESCRIPTION

shapes is a program for numerical simulation of electromagnetic waves and pulses scattering on metal and dielectric objects of various shapes. In the current version of **shapes** (2.0) the simulated system is restricted to 2 dimensions (x, y) and the simulated wave is (E_x, E_y, H_z).

Several features distinguish **shapes** from other programs of this type. First, **shapes** can be run in parallel on a CPU cluster under MPI, but it can be run sequentially too. Second, **shapes** is a multigrid program that lets you refine space and time resolution in places of special interest. Third, **shapes** can do all its IO in parallel, writing HDF5 files preferably on a parallel file system such as PVFS or GPFS. These can be then viewed and postprocessed with a special tool called **chombovis**(1). Fourth, **shapes** takes its input from a simple human-readable text file. Complex object shapes can be described in terms of multiple rectangles, circles and triangles, each of which can be assigned different material properties. The figures can overlap and can be additionally altered by super-imposing masks. Multi-grids can be defined locally in terms of similar constructs that build on multiple rectangles, circles, triangles and masks.

shapes implements its computations using finite difference time domain method in combination with the total-field/scattered-field algorithm and perfectly matched layer (PML) absorbing boundary conditions (ABCs) on the periphery of a rectangular computational domain. Metals are described in terms of a combined Drude/Lorentz model with an arbitrary number of resonances.

shapes can output various types of data. Each of the computed fields, E_x, E_y, and H_z can be output, as well as metal distributions, energy density and spectral response. Multi-grid layout is always output on the generated HDF5 files. The sequential version of **shapes** can output data in the Gnuplot format too, but this is disabled in the parallel MPI version and input file directives that pertain to Gnuplot output are ignored in this case. In the HDF5 mode, all fields are collated on a single file. In the Gnuplot mode, each field is output on a separate file.

shapes is built on top of the **Chombo**(3) toolkit for adaptive mesh refinement (AMR) programming and is made of two separate components, a C++ shell written in Chombo and a set of Fortran-77 subroutines and functions that implement the physics of the simulation. The C++ shell manages data flow between computational processes and levels of the multigrid. Chombo takes care of all input, output and parallelization. The Fortran-77 subroutines can be used in a stand-alone Fortran program, either a sequential one, or a parallel one, without change. An example of such a program is provided with the source.

It should be relatively easy even for a novice *Chombo programmer* to substitute the Fortran-77 subroutines provided currently with different ones. The code is sufficiently modular for this.

The vision for this project is that down the road utilities and framework will be provided to make this possible to a casual user of **shapes** too. But this is less trivial. To begin with input data flows in a fairly complicated way through the Chombo C++ shell to the Fortran subroutines. Furthermore, these and other Chombo C++ functions interact with each other recursively, since this is how the multi-grid algorithm is implemented.

OPTIONS

Every keyword used in the **shapes** input file, see **shapes**(5) for more information, can be used as an option too. For example, instead of specifying **level0.time = 0.0** on the input file, the same can be done on the command line, e.g.,

mpirun shapes level0.time = 0.0 *shapes-input-file*

but this is not a recommended practice. I haven't tested it for starters, and it is good to have a full written record of all options and parameters declared for the run on the **shapes** input file too. The options and parameters specified for the run are written on both Gnuplot and HDF5 files as headers.

USAGE

shapes was developed for use in the batch mode. The user is expected to provide an input file, see **shapes**(5), that characterizes the system that is to be simulated. The job is then submitted to PBS. The PBS script should take care of obtaining a sufficient number of nodes, creating the data directory and changing to it prior to invoking MPI for the run.

The sequential version of **shapes** can be run interactively or under PBS in a high capacity (as opposed to the high capability parallel execution) parameter space exploration mode. It can be used to test a given configuration on a low resolution grid too.

The job will generate multiple output files in the HDF5 or Gnuplot format. The files will contain field data for the time slices at which they will have been dumped. The frequency of making such snapshots is regulated by setting appropriate parameters on the input file.

The way data is written on the HDF5 files is Chombo specific and a special tool, **chombovis**(1), is needed to view and further process the data.

It is best to dump the HDF5 data on a parallel file system, e.g., PVFS or GPFS, especially if there is going to be a lot of it. It is also best to run postprocessing, e.g., a movie extraction or inspection of specific time slices, directly from the parallel file system without transferring the data to other systems. Because ChomboVis is a rather slow python script that talks do your display using a VTK toolkit and OpenMesa -- all of which involves a lot of computation, you should not run it on one of the two front end nodes, i.e., **jlogin1** or **jlogin2.** Instead you should request an interactive PBS job (with **qsub -I**), you should then set your **DISPLAY** variable on the allocated node, allow access to your X11 server, and invoke chombovis on the allocated node. The same applies to extraction of animations (also with ChomboVis) from the HDF5 data files.

In my experience you may actually get a better ChomboVis performance this way than if you were to transfer the data to your desktop workstation and then run ChomboVis on it.

The Gnuplot style output is a normal text file that is human readable. It is organized so that it can be displayed with Gnuplot using the **splot** command.

Both HDF5 and Gnuplot files can be postprocessed to generate GIF animations. This is extremely painful and slow with the HDF5 system at present, but very easy with the Gnuplot system.

EXAMPLES

You will find example input files in
/soft/apps/packages/photonic-packages/Confs
and example PBS scripts invoking **shapes** in
/soft/apps/packages/photonic-packages/PBS.
Example outputs can be found in
/soft/apps/packages/photonic-packages/var/run

FILES

/soft/apps/packages/photonic-packages/bin/shapes – this is an easy to invoke script that sets the environment and then invokes the *sequential* version of the code.

/soft/apps/packages/photonic-packages/bin/pshapes – is an easy to invoke script that sets the environment and then invokes the *parallel* version of the code.

/soft/apps/packages/photonic-packages/bin/shapes2d.Linux.g++.g77.ex – this is the actual sequential binary.

/soft/apps/packages/photonic-packages/bin/shapes2d.Linux.g++.g77.DEBUG.ex – this is the sequential binary with debugging turned on.

/soft/apps/packages/photonic-packages/bin/shapes2d.Linux.mpiCC.ifort.MPI.ex – this is the parallel MPI binary.

/soft/apps/packages/photonic-packages/bin/shapes2d.Linux.mpiCC.ifort.DEBUG.MPI.ex – the parallel MPI binary with debugging turned on.

/soft/apps/packages/photonic-packages/src/Shapes-2.0 – this is where the latest source to this version of the program lives.

/soft/apps/packages/photonic-packages/Chombo-1.4-24Nov2005 – this is where the actual Chombo libraries live against which this version of **shapes** has been linked.

/soft/apps/packages/photonic-packages/hdf5-1.6.5 – this is where sequential HDF5 libraries and utilities live against which this version of **shapes** has been linked.

/soft/apps/packages/photonic-packages/phdf5-1.6.5 – this is where parallel HDF5 libraries and utilities live against which this version of **shapes** has been linked.

ENVIRONMENT

shapes depends on various dynamic libraries. These creep into **shapes** from HDF5, Intel C++ and Fortran compilers and MPI. **shapes** is invoked through scripts that should set *LD_LIBRARY_PATH* and *LD_RUN_PATH* variables to pick up the right libraries.

DIAGNOSTICS

shapes is compiled for production. In this mode Chombo debugging is switched off. Otherwise debugging slows the execution of the program considerably. This means that when an error condition is encountered Chombo subroutines will keep going returning nonsensical results. The really dangerous and peculiar feature of Chombo is that the program may *not* crash, sic! **shapes** itself traps numerous error conditions. These are flagged, but the program currently is not designed to abort.

At this preliminary stage little effort has been invested in making the program *foolproof* (it is just an expression, it does not imply disrespect for the prospective users). On the other hand a lot of effort *was* invested in ensuring that the program converts correctly formulated input into correct output.

Normally **shapes** runs silently, but it is possible to make it very chatty. There are seven parameters in the *Chat* group of the input file that regulate **shapes'** verbosity. These are

chat.print_versions – when set to 1, it makes **shapes** print RCS versions of every module it is made of. The author of the code, i.e., me, is printed too.

chat.chombo_verbose – when set to 0, it stops **shapes** C++ routines from announcing themselves. It can be set to an integer greater than 0 to make them talk. Setting *chombo_verbose* to anything other than zero activates the *chat.print_actions* option too.

chat.fortran_verbose – when set to 0, it stops **shapes** Fortran routines from announcing themselves. It can be set to an integer greater than 0 to make them talk. For parallel runs, setting *chat.fortran_verbose* to anything other than 0 will result in an insane amount of output.

chat.print_level_0_domain – when set to 1 it makes **shapes** print information pertaining to the level 0 domain that has been constructed for the run. Set it to 0 if you don't want this output.

chat.print_levels – when set to 1 it makes **shapes** print information pertaining to every level that has been constructed during the run. Activating this parameter activates *chat.print_level_0_domain* too. Set it to 0 if you don't want this output.

chat.print_min_max – when set to 1 it makes **shapes** print minima and maxima of fields every time an image is dumped. The values are printed for this current time slice and for the run as a whole up to this point in time. Set it to 0 if you don't want this output.

chat.print_actions – when set to 1 it makes the **shapes** functions print everything they do including time stamps. This can be used to observe the recursion, synchronization and data movements between the levels.

When **shapes** runs in parallel under MPI, each process writes its diagnostic output on *pout.X* where *X* is the process number.

Apart from these parameters, every Chombo function can be made to talk on its own by setting an appropriate *watch* flag. For example, in order to make function *full_copy* talk, set *watch.full_copy* to 1 or a higher integer number. Setting it to a number greater than 2 activates Fortran verbosity for Fortran subroutines called from within *full_copy* too.

NOTES

It is preferable to dump HDF5 files on a parallel file system, but if such is not available, NFS will do just fine, however slow it may be, assuming, of course, there is enough space on it.

Extraction of animations from HDF5 files with Chombovis is a major headache.

BUGS

This section documents known *design* bugs rather than specific implementation bugs of the kind that may make the program crash or return incorrect results. To the best of my knowledge at this juncture, the program doesn't have such bugs. This, of course, does not mean that it really doesn't have any serious bugs of this nature. It only means, I don't presently know of any. One does not normally *document* bugs in this category. One *fixes* them as soon as possible. The *documented* bugs are the ones one can live with.

shapes is at an early stage still and it is going to evolve in the 3D direction. But its 2D version is quite useable now, hence the decision to freeze its development and release it, even though not everything may be quite buttoned up.

Let me just list some design bugs that I know of and that I will attempt to fix in future versions of the code.

shapes depends on too many dynamic libraries. This is the result of building the program on top of a prototype environment, namely, Chombo, but also the result of certain default settings in Linux, MPI, HDF5 and compilers used to produce the binary. It may be possible to fix this down the road. A statically linked binary would be preferable for performance reasons. It would make the distribution of the code easier too.

Input can be provided in *natural* units only. This is because **shapes** computes everything in natural units. It is usually preferable to do so - computations can be implemented more efficiently and accurately. A detailed procedure for conversion of input data from SI or CGS units to natural units is discussed in the User Guide.

The input file can be quite difficult to construct, especially if multiple media are distributed in a complex manner. A graphic tool could be provided to generate the input file automatically. We may consider doing this in the next phase of the project, where we would focus on nano-device optimization rather than on the development of the actual simulator.

It is common for explicit time stepping programs to develop noise at some stage, whereupon the computation diverges rapidly. Multigrid exacerbates the problem, because it is itself an additional source of noise. I have looked at various ways of alleviating this problem and something may be provided in future versions of the code. A more insightful mathematical analysis of what happens on the multigrid boundaries will be needed.

Even with multiple Drude/Lorentz resonances **shapes** will not account for non-linear and quantum effects in the media. These may be added to future versions of the program.

shapes is a 2D program. It is going to evolve into a fully-featured 3D simulator.

There is no provision for saving the state of the job and restarting the job at present.

shapes does not do enough input checking to ensure the robustness of the program. When compiled for production, it may continue to run even when errors are encountered and flagged.

VERSION

This page documents the January 12, 2006 version of **shapes.** This version is probably the last major 2D version of the program. Future development will focus on 3D simulations.

AUTHOR

At present **shapes** is a product of one author only, Zdzislaw (Gustav) Meglicki of Indiana University (*gustav@indiana.edu*), with occasional assistance of his cats, Bambosz and Sofa, who tend to type random strings into the source files in unexpected places. This is going to change as the application develops and becomes incorporated into a nano-photonics toolkit yet to be constructed.

SEE ALSO

shapes(5), **mpirun**(1), **chombo**(3), **chombovis**(1), **dump_fromstate**(1), **dump_ppms**(1), **makemovie**(1), "Shapes User Guide".